Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro zálohování konfigurací aktivních síťových prvků.

System for backup of network devices configuration.

2014 **Lukáš Topiarz**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Topiarz**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém pro zálohování konfigurací aktivních síťových prvků
System For Backup of Network Devices Configuration

Zásady pro vypracování:

Navrhněte a realizujte systém, který bude udržovat zálohy aktuálních konfigurací síťových prvků. Systém bude obsahovat správu verzí a také nástroje pro jednoduchou obnovu konfigurace. Taktéž bude umožňovat možnost porovnání konfigurací v různých časech.

1. Vytvořte virtuální testovací topologii s více síťovými prvky pomocí software GNS3.
2. Implementujte systém pro pravidelné stahování konfigurací na server.
3. Implementujte možnost stažení konfigurace mimo pravidelné zálohy.
4. Použijte vhodný systém pro správu verzí. Vytvořte nástroje pro jednoduché porovnávání verzí konfiguračních souborů.
5. Navrhněte vhodné zabezpečení na úrovni přístupových práv.
6. Implementujte i možnost zálohování obrazu operačního systému použitého v aktivním prvku.

Seznam doporučené odborné literatury:

[1] Evi, NEMETH, Garth SNYDER a Trent R. HEIN. Linux: kompletní příručka administrátora. 1.vyd. Brno: Computer Press, 2004, 828 s. ISBN 80-722-6919-4
[2] WRZESZCZ, Zdisław. TCL/TK: podrobný průvodce programovacími jazyky. Vydání první. Brno: Computer Press, 2005, 388 s. ISBN 80-251-0781-7
[3] LIBES, Don. Exploring expect: a tcl-based toolkit for automating interactive programs. 1st ed. Sebastopol, CA: O'Reilly, c1995, xxxiii, 566 p. ISBN 15-659-2090-2

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Těrlicku dne: *5. května 2014*

podpis studenta

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce, panu Ing. Davidu Seidlovi, Ph.D., za pomocné rady a konzultace při tvorbě této práce.

Abstrakt

Cílem této práce je navrhnout a realizovat systém, který bude automaticky stahovat a udržovat zálohy konfigurací aktivních síťových prvků od výrobce Cisco s operačním systémem Cisco IOS.

Systém bude obsahovat nástroje pro zálohu, ale také pro správu verzí konfigurací a možnost jejich jednoduché obnovy. Systém bude dále umožňovat porovnávání konfigurací, zálohových v různých časech. Systém bude umožňovat i zálohu binárního obrazu operačního systému síťového prvku.

Proces zálohování síťových prvků bude moci být spouštěn jak automaticky, tak ručně.

Veškeré parametry síťových zařízení, způsoby komunikace a cesty k adresářům budou načítány z jednotného konfiguračního souboru.

Klíčová slova

automatické zálohování konfigurace; zálohování Cisco IOS; Cisco; IOS; porovnání konfigurací; obnova konfigurace

Abstract

Aim of this thesis is to design and realiaze system for automatic backup and keeping of configuration files from active network devices of Cisco vendor, which have Cisco IOS operation system.

Except of tools for backup, system will also consists of tools for version keepeing and possibility of simple configuration recovery. It will also allow comparision of configuration files backed up in different times. System will also make possible backup of network devices´s binary operating system file.

Process of backuping can be started automatically or manually.

All network devices´s parameters, ways of communication and directory paths are loaded from single configuration file.

Key words

automatic configuration backup; Cisco IOS backup; Cisco; IOS; configuration comparison; configuration recovery

Seznam použitých zkratek

Zkratka	Význam
ASA	Adaptive Security Appliance
CLI	Command Line Interface
DEB	archiv balíčkovacího systému distribuce Debian
GZIP	GNU zip
GNU	GNU's Not Unix
IOS	Internetwork Operating System
NVRAM	Non-Volatile Random-Access Memory
OS	Operating Systém
POST	Power On Self Test
RAM	Random-Access Memory
ROM	Read-Only Memory
RPM	Red Hat Package Manager
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TAR	Tape archiver
TCL	Tool Command Language
TFTP	Trivial File Transfer Protocol
VTY	virtual serial interface

Obsah

Úvod - 9 -
1 Popis platformy - 10 -
1.1 Operační systém Cisco IOS - 10 -
1.2 Možnosti připojení k zařízení s OS Cisco IOS - 10 -
1.3 Vlastnosti CLI systému Cisco IOS - 11 -
1.4 Typy konfigurací a zavedení binární IOS obrazu - 12 -
Analýza konkurenčních řešení - 13 -
1.5 Seznam existujících řešení - 13 -
1.5.1 Ruční správa a nastavení v CLI jednotlivých zařízení - 13 -
1.5.2 Kron - 14 -
1.5.3 Nekomerční software Rancid - 14 -
1.5.4 Komerční software Network Configuration Manager - 15 -
1.6 Srovnání existujících řešení - 16 -
1.7 Specifikace vlastního cíle - 16 -
1.7.1 Shrnutí požadavků - 16 -
2 Návrh vlastního řešení - Ciscoback - 17 -
2.1 Funkční požadavky - 17 -
2.2 Obecný princip - 17 -
2.3 Komponenty - 18 -
2.3.1 Operační systém - 18 -
2.3.2 Řídící program, programovací jazyk TCL - 18 -
2.3.3 Síťová komunikace - 19 -
2.3.4 Periodické spouštění - 19 -
2.3.5 Logování - 19 -
2.3.6 Konfigurační soubor - 19 -
2.3.7 Verzovací software - 19 -
2.4 Systémové a síťové požadavky - 20 -
2.4.1 Systémové požadavky na server - 20 -
2.4.2 Síťové požadavky na server - 20 -

2.4.3 Požadavky na konfigaraci Cisco IOS ..... - 20 -
3 Realizace ..... - 21 -
3.1 Řídící program ..... - 21 -
3.1.1 Princip automatizace pomocí Expect ..... - 21 -
3.1.2 Funkce pro připojení k síťovému zařízení ..... - 22 -
3.1.3 Funkce pro vyvolání zálohy na tftp ..... - 23 -
3.1.4 Funkce pro GIT add a commit ..... - 24 -
3.1.5 Funkce pro rollback konfigurace ..... - 25 -
3.1.6 Funkce pro kontrolu syntaxe konfiguračního souboru ..... - 26 -
3.1.7 Rozsah zdrojového kódu ..... - 28 -
3.2 Init skript ..... - 28 -
3.3 Konfigurační soubor ..... - 28 -
3.4 Log ..... - 29 -
3.5 Manuálová stránka ..... - 29 -
3.6 Deb balíček ..... - 30 -
4 Testovací prostředí ..... - 31 -
4.1 Virtuální prostředí pomocí GNS3 ..... - 31 -
4.1.1 GNS3 ..... - 31 -
4.2 Testování na fyzických topologiích ..... - 32 -
5 Dostupnost ..... - 33 -
Závěr ..... - 34 -
Použitá literatura ..... - 35 -
Seznam příloh na CD ..... - 36 -

Úvod

V době, kdy díky rozvoji Internetu, takřka každý počítač, notebook, mobilní zařízení a další zařízení, vyžadují připojení k síti, roste vysokým tempem množství řídících síťových prvků. S narůstajícím počtem síťových prvků se již síťoví administrátoři nejsou schopni z časových a ekonomických důvodů věnovat ruční konfiguraci, správě a zálohování nastavení každého zařízení zvlášť. Nastupuje automatizace úloh s cílem zvýšení počtu spravovaných síťových zařízení za stejný čas s minimálními ručními zásahy.

Tato práce se věnuje problematice automatické zálohy síťových zařízení, jakými jsou především přepínače a směrovače. V každé síťové topologii je přinejmenším vhodné zálohovat a udržovat konfigurační soubory síťových prvků. S narůstající velikostí topologie tato vhodnost přechází v nutnost. V případě selhání zařízení umožní zálohovaná konfigurace rychlejší nasazení nového prvku. Pokud nastane na zařízení problém v důsledku změny konfigurace, může administrátor porovnávat konfigurace na běžícím prvku s předešlými, správně fungujícími verzemi.

Mou hlavní motivací a cílem je vytvoření instalovatelného deb balíčku pro operační systému GNU/Linux odvozený od distribuce Debian, který bude provádět automatickou zálohu konfigurací a binárního obrazu operačního systému síťových prvků od společnosti Cisco Systems, Inc., na kterých funguje operační systém Cisco IOS. Systém, který navrhuji je určen pro jednotky až desítky zálohovaných zařízení v síti.

Systém, který navrhuji, vznikl z nutnosti správy sítě ve společnosti, ve které pracuji jako síťový administrátor. Mým cílem je také vytvořit tento systém natolik obecně, aby byl lehce přenositelný i na jiné síťové topologie a pro jiné síťové administrátory.

1 Popis platformy

Cílovými síťovými prvky pro zálohování jsou zařízení od společnosti Cisco Systems, Inc. Jedná se enterprise modely přepínačů a směrovačů, ale také některé bezpečnostní appliance typu Cisco ASA a bezdrátové přístupové body Cisco Aironet.

Společným znakem těchto zařízení je operační systém Cisco IOS, či systémy se shodnou syntaxí a funkčností týkající se možnosti zálohování a přístupu ke správě zařízení. Podobným systémem jako Cisco IOS je například Cisco ASA v bezpečnostních appliancích Cisco ASA.

1.1 Operační systém Cisco IOS

V současných enterprise přepínačích a směrovačích od výrobce Cisco Systems, Inc. je používán operační systém Cisco IOS. Jeho nejaktuálnější verze nese označení 15.3S. Jedná se o multitaskingový operační systém s integrovanými směrovacími, přepínacími a telekomunikačními funkcemi. [4]

Do roku 2006 byl operační systém Cisco IOS monolitickým systémem, kde všechny procesy sdílely společný paměťový prostor. Neexistovala žádná ochrana mezi procesy, které mohly v případě chyby ohrozit data jiného procesu. Od roku 2006 je operační systém plně modulární, multitaskingový, s možností odděleného paměťového prostoru pro jednotlivé procesy, možností restartovat selhané procesy a možností preempce vykonávaných procesů.

V současných síťových prostředích existují nejčastěji zařízení s OS ve verzi z rodiny 12.4.x (První verze 12.1E uvedena 15. 9. 2006.) a novou verzí z rodiny 15.x (Uvedeno na trh 10 4. 2009.). Tato bakalářská práce bude pracovat pouze s verzí Cisco IOS 12.2 a vyšší.

Jednotlivé systémy v rámci stejné produktové řady se dále liší sadou funkcí, či balíčky. Směrovače mají na výběr typicky z osmi sad funkcí, či balíčků a přepínače z pěti možných. Sady funkcí se liší svým zaměřením. Např. sada funkcí pro zabezpečení, nebo sada pro další datové protokoly.

1.2 Možnosti připojení k zařízení s OS Cisco IOS

Zařízení je možné spravovat pomocí CLI, webového přístupu, či pomocí snmp protokolu.

Pro účely mé práce jsem zvolil možnost správy přes příkazovou řádku. Možnosti připojení k zařízení přes CLI jsou dvě. Připojení zabezpečené pomocí SSH protokolu, nebo připojení nezabezpečené pomocí protokolu Telnet. Mnou implementované řešení bude schopno komunikovat se síťovými zařízení obojím způsobem. Pro zabezpečené připojování budu používat SSH protokol ve verzi 2.

1.3 Vlastnosti CLI systému Cisco IOS

Systémy Cisco IOS obsahují pro svou správu a konfiguraci sadu příkazů používaných v příkazové řádce. Systém je rozdělen na 16 (úroveň 0-15) bezpečnostních úrovní a 4 příkazové módy. Práva na spouštění příkazů se odvíjejí od úrovně, v níž se v CLI operačního systému uživatel nachází.

Bezpečnostní úrovně mohou být libovolně rozděleny mezi příkazové módy. Příkazové módy se dělí na:

- Uživatelský mód (**user EXEC**) - Výchozí mód po přihlášení. V CLI se zobrazuje příkazová řádka jako:

```
NázevZařízení>
```

- Privilegovaný mód (**privileged EXEC**) - Výchozí mód pro přestup do dalších konfigurací. Má vyšší práva a umožnuje tedy zobrazit více údajů nežli user EXEC mód. V CLI se zobrazuje příkazová řádka jako:

```
NázevZařízení#
```

- Globální konfigurační mód (**global configuration**) - Mód pro konfigurace funkcí, které ovlivňují celý systém. V CLI se zobrazuje příkazová řádka jako:

```
NázevZařízení(config)#
```

- Konfigurace interfacu (**interface configuration**) - Mód pro konfiguraci vlastností určitého interface. V CLI se zobrazuje příkazová řádka jako:

```
NázevZařízení(config-if)#
```

Pro přestup z výchozího módu po přihlášení do privileged Exec módu slouží příkaz:

```
enable
```

Pro přestup z privileged Exec módu do global configuration slouží příkaz:

```
configure  terminal
```

Pro přestup z global configuration módu do konfigurace interface slouží příkaz:

```
interface {jméno interface}
```

1.4 Typy konfigurací a zavedení binární IOS obrazu

Po zapnutí zařízení se spouští mikrokód z ROM paměti. Po provedení POST (Power On Self Test) je hledán obraz IOSu standardně ve Flash paměti. Celý IOS je uložen v jednom image souboru s příponou bin, který je zpravidla uložen na Flash paměti zařízení. Pokud je validní obraz nalezen, pak je zaveden. Pokud ne, pak může být dále hledán na síti a zaváděn pomocí tftp bootu, nebo zařízení v případě neúspěchu nastartuje do omezené funkce RxBoot.

Konfigurace, kterou si zařízení načítá při bootování do running-config, se nazývá **startup-config.** Tuto konfiguraci zařízení při bootování hledá v NVRAM. Pokud v NVRAM konfigurace chybí, pak zařízení vyvolává express setup, jež nastavuje výchozí kofiguraci.

Konfigurace, podle níž se zařízení za běhu řídí, se nazývá **running-config** a je uložená v RAM paměti zařízení. Veškeré zadávané příkazy se okamžitě provádějí a ukládají se do této konfigurace.

Analýza konkurenčních řešení

Jak jsem již naznačil v úvodu, tato oblast není v problematice počítačových sítí ničím novým, a tudíž existují již hotová řešení. V následujících kapitolách chci tato řešení představit, srovnat jejich klady a zápory a obhájit potřebu implementace mého vlastního řešení.

1.5 Seznam existujících řešení

1.5.1 Ruční správa a nastavení v CLI jednotlivých zařízení

Cisco IOS má několik příkazů, jež umožňují ručně, či automatizovaně zálohovat konfigurace a binární obraz systému. Pro volání příkazů je nutné přejít do privileged EXEC módu. Nevýhodou tohoto přístupu je nutnost provádět dané příkazy, nebo nastavení alespoň jednou ručně na každém síťovém prvku zvlášť. Porovnávání konfigurací se poté provádí příslušnými příkazy z CLI zařízení.

1.5.1.1 Jednorázové zálohování

Pro jednorázové uložení running-config či startup-config na tftp server slouží příkaz:

```
DEVICE#copy <running-config|startup-config> <tftp:|ftp:|scp:>
```

Pro jednorázové zkopírování binárního souboru obrazu operačního systému lze použít příkaz:

```
DEVICE#copy flash:<název ios image> <tftp:|ftp:|scp:>
```

1.5.1.2 Plánované zálohování pomocí funkce archive

Výše uvedené jednorázové zálohování lze zautomatizovat vytvořením předvolby zálohovacího umístění a také naplánováním pravidelného spouštění pomocí konfiguračních archivů. Konfigurační archivy nabízí mechanismy pro ukládání a organizaci konfiguračních souborů systému. Konfiguraci je nutno provést v global configuration módu.

```
DEVICE(config)#archive
DEVICE(config-archive)#path tftp://<ip adresa>/<cesta a název
souboru na tftp serveru>
```

Následně příkazem z privileged EXEC módu lze spustit zálohu running-config konfigurace.

```
DEVICE#archive config
```

Automatické provádění archivace po časovém intervalu lze nastavit v global configuration módu pomocí:

```
DEVICE(config-archive)#time-period <čas v minutách>
```

Pomocí příkazu archive lze také zálohovat obraz IOSu na server. Obraz je před uploadem zabalen do tar archivu.

```
DEVICE#archive upload-sw <ftp:|tftp:><cesta a případné
přihlašovací údaje pro ftp>
```

Konfigurace lze pomocí archive také porovnávat. Např. porovnání uložené konfigurace na serveru s running-config:

```
DEVICE#show archive config differences tftp://<ipadresa>/<cesta>
system:running-config
```

Ukázky použití příkazu archive jsem čerpal z webu www.samuraj-cz.com. [5]

1.5.2 Kron

Kron je zabudovaný plánovač úloh v Cisco IOS. Umožnuje naplánování spouštění CLI příkazů při startu zařízení, v pravidelných intervalech, či v definovaných časech. [7]

Pro použití kronu je nutné nadefinovat kron policy-list. Příkazy uložené v tomto listu jsou poté prováděny řádek po řádku. K tomuto listu se poté nastaví čas, kdy má být seznam příkazů vykonán. Pro provoz kronu je tedy nutné mít na zařízení správně nastaven systémový čas. Kron je nastavován v global configuration módu.

Ukázka jednoduchého policy-listu pro zálohování startup-config:

```
DEVICE#enable
DEVICE#configure terminal
DEVICE(config)#kron policy-list <název policy-listu>
DEVICE(config-kron-policy)#cli show startup-config | redirect
tftp://<ip adresa>/<cesta>
```

Ukázka naplánování spouštění policy-listu každou neděli ve 23:00:

```
DEVICE(config)#kron occurrence <název opakované úlohy> at 23:00
Sun recurring
DEVICE(config-kron-occurrence)#policy-list <název policy-listu>
```

Z podstaty kronu je tento přístup omezen pouze na spouštění neinteraktivínívh příkazů.

1.5.3 Nekomerční software Rancid

Rancid, vyvíjený společností Shrubbery Networks, Inc., je nekomerční software určený pro zálohu a verzovaní konfiguraci enterprise přepínačů a směrovačů Cisco, Juniper, HP

a dalších. Licence softwaru je zcela nekomerční. Aktuální verze Rancid je 3.0. Informace čerpány z webových stránek projektu Rancid od společnosti Shrubbery Networks, Inc. [7]

Ke svému fungování vyžaduje možnost vzdáleného připojení k CLI síťových prvků (SSH, či telnet). Pro správu verzí záloh používá verzovací systém CVS, nebo jeho nástupce Subversion. Pro komunikaci se síťovými zařízeními používá rozšíření skriptovacícho jazyka Tcl Expect.

Rancid je ovládán a nastavován z příkazové řádky. Systémovými požadavky pro jeho běh jsou operační systém GNU/Linux, FreeBSD, Solaris a OS X.

1.5.4 Komerční software Network Configuration Manager

Network Configuration Manager (NCM) od společnosti Solarwinds umožňuje centralizovanou správu velkého množství síťových prvků různých výrobců. Podporovány jsou síťové prvky, k nímž se lze vzdáleně připojit k jejich CLI (SSH, telnet). Výchozím bodem pro správu, změny konfigurací, nastavování a spouštění zálohování, je webové rozhraní. Tento software je zatížen placenou licencí. Aktuální verze NCM je 7.2. Informace čerpány z webových stránek projektu NCM od společnosti Solarwinds. [6]

Jednou z komponent je automatické zálohování konfigurací síťových prvků. NCM umožňuje jak ruční, tak plánované zálohování. Mezi další funkčnosti patří uchovávání verzí konfiguračních souborů, možnost prohlížení změn, shlukování zařízení do logických skupin (Např. podle lokace, výrobce, oddělení a dalších uživatelsky definovatelných vlastností.).

Systémové požadavky NCM vůči hardware (3GHz dual-core CPU, 3GB RAM, 20GB diskový prostor) jsou vzhledem k dnešním parametrům serverů zanedbatelné. Zajímavým aspektem jsou však požadavky na softwarové vybavení. NCM vyžaduje OS minimálně Microsoft Windows 2003 SP2, .Net framework alespoň ve verzi 3.5 SP1 a SQL databázi minimálně SQL Server 2005 SP1.

Zřejmou nevýhodou tohoto řešení je tedy kombinace pořizovacích cen za hardware pro server, licence pro produkty Microsoft a licence pro samotný Network Configuration Manager (Nejméně 2.195 Euro pro zalicencování produktu pro 50 spravovaných zařízení.) [6]

1.6 Srovnání existujících řešení

Ruční zálohování z CLI je nejjednodušší přístup k této problematice. Je podporován každým zařízením.

Vlastnosti plánovače úloh Kron se mohou lišit podle sady funkcí v IOSu. Je tedy nutné dbát na omezení, které daná verze IOSu může způsobit. Použitím plánovače je prvním krokem k automatizaci práce síťového administrátora.

První dva zmiňovaná řešení mají však společnou nevýhodu a sice nutnost jednotlivě nakonfigurovat každý síťový prvek. V případě použití Kronu je nutné při změně naplánovaného času, či změně serveru, který slouží k ukládání záloh, překonfigurovat každý prvek. Tato řešení tedy nejsou lehce škálovatelná a nehodí se do větších síťových topologií.

Centralizovaná řešení mimo síťové prvky jsou nutnou volbou pro větší síťové topologie. Výhodou těchto řešení je lehká škálovatelnost a flexibilita při změnách v zálohovací politice. Společnou vlastností těchto řešení je komunikace se síťovými prvky pomocí protokolů SSH, či telnet. Rozdíly mezi software NCM a Rancid jsou především ve dvou bodech. NCM je robustnější řešení pro hromadnou správu, nikoliv pouze pro zálohování a správu verzí konfiguračních souborů. NCM vyžaduje finančí vklad do licencí samotného softwaru tak i do operačního systému, na němž funguje. Rancid je oproti tomu poskytován nekomerčně.

1.7 Specifikace vlastního cíle

Mnou navrhované řešení pro prostředí v řádů desítek zálohovaných zařízení se nejvíce blíží softwaru Rancid. Principy, na nichž funguje jeho komunikace se síťovými prvky a verzování konfigurací, jsou totožné s mým řešením.

Výhodou vlastní implementace řešení, podobné Rancidu, je vyšší přizpůsobitelnost na míru síťové topologie, jež jako síťový administrátor spravuji.

1.7.1 Shrnutí požadavků

- Systém pro zálohování running-config, startup-config, IOS obrazu.
- Plánováné i manuální spouštění záloh z prostředí serveru.
- Možnost obnovy konfigurace ze serveru.
- Centrální konfigurační soubor pro popis všech zálohovaných zařízení.
- Verzování a možnost prohlížení změn v konfiguračních souborech.

2 Návrh vlastního řešení - Ciscoback

V následujících kapitolách popíšu funkční a nefunkční požadavky na mou práci a pokusím se přiblížit principy fungování a vybrané technologie pro realizaci mého zálohovacího řešení. Výsledné řešení jsem se rozhodl pojmenovat jako **Ciscoback**.

2.1 Funkční požadavky

- Systém umožní plánované, pravidelné zálohovaní konfigurací a obrazů operačního systému síťových prvků Cisco.
- Systém umožní ruční stažení konfigurace, či obrazu operačního systému.
- Systém bude spravovat verze zálohovaných konfiguračních souborů a umožní jejich porovnávání.
- Systém umožní obnovu konfigurace síťového prvku.
- Systém bude dostupný jako instalovatelný deb balíček.

2.2 Obecný princip

Po nainstalování balíčku bude na serveru existovat řídící program, init skript a konfigurační soubor. Veškeré informace o zálohovaných zařízeních, adresářích používaných tftp démonem a časech spouštění zálohy budou uloženy v konfiguračním souboru. Primárním ovládacím bodem pro plánované zálohování bude init skript. Přes init skript se zaregistrují úlohy do systémového crontabu. Podle záznamů v crontabu pak bude Cron spouštět řídící program s parametry rozlišujícími mezi zálohováním konfiguračních souborů a zálohováním Cisco IOS obrazu síťových prvků. Řídící program lze spouštět i ručně s příslušnými parametry.

Úlohou řídícího programu je především automatizace interaktivního přihlášení k síťovým prvkům, zadávání přihlašovacích údajů a příkazů pro kopírování na tftp server. Pokud bude zálohovat konfigurační soubor síťového prvku, pak po dokončení spustí operaci verzovacího software nad adresářem, kde uložil konfiguraci.

Pro prohlížení rozdílů v konfiguračních souborech slouží verzovací systém, který je volán přes řídící program.

Aplikování změn v zálohování po úpravě konfiguračního souboru, nebo zrušení pravidelného zálohovaní se provádí primárně přes init skript.

2.3 Komponenty

2.3.1 Operační systém

Jako operační systém pro instalaci a běh Ciscobacku jsem zvolil GNU/Linux distribuce odvozené od OS Debian.

Debian GNU/Linux je OS vytvářený komunitou okolo projektu Debian. Skládá se z jádra Linux a sady základních nástrojů a programů pocházejících z projektu GNU [9].

Zvolil jsem ho z následujících důvodů:

- OS zdarma.
- Univerzálnost, velké množství nástrojů a programů.
- Nízká hardwarová náročnost.
- Množství podporovaných počítačových platforem.
- Široké zastoupení a podpora.

2.3.2 Řídící program, programovací jazyk TCL

Pro komunikaci se síťovými prvky, spouštění ostatních softwarových nástrojů v OS, načítání dat z konfiguračního souboru, registraci úloh v crontabu a jiné řídící konstrukce jsem zvolil použití programu napsaného v interpretovaném programovacím jazyku Tcl.

Pro samotné zautomatizování připojení a provedení interaktivní komunikace se síťovými prvky jsem zvolil rozšíření jazyka Tcl Expect.

Pro spouštění, zastavení a načtení informací do zálohovací služby použiju init skript napsaný jako shellovský skript pro shell Bash.

Kombinaci interpretovaného jazyka Tcl a jeho rozšíření Expect jsem zvolil z důvodů rychlé a pohodlné implementace dostačující pro potřeby zadání. Vyhnul jsem se nutnosti programovat veškeré základní konstrukce jako např. implementace síťové komunikace apod. Využil jsem již existující funkce a technologie a složil z nich logický celek pro potřeby mého cíle.

2.3.2.1 TCL

Tcl je interpretovaný programovací jazyk. Textovým intepretem je tclsh. Mezi některé vlastnosti patří nízká hardwarová náročnost, veškeré operace jsou příkazy (Imperativní jazyk.), všechny datové typy jsou reprezentovány jako textové řetězce. [2]

2.3.2.2 EXPECT

Expect je rozšířením jazyka Tcl. Jedná se o program určený pro zautomatizování interakce s jinými programy, které využívají textové terminálové rozhraní. Expect nahrazuje ruční zadávání textu uživatelem a vytváří automatizovanou interakci. [3]

2.3.3 Síťová komunikace

Program implementovaný pomocí Tcl a rozšíření Expect bude spouštět a ovládat programy pro komunikaci se síťovými prvky. Připojení k síťovým prvkům bude možné přes protokol ssh a telnet a proto budou k tomuto využity stejnojmenné programy, které poskytují softwarové klienty telnet-client a ssh-client.

2.3.4 Periodické spouštění

Pravidelné spouštění řídícího skriptu s příslušnými parametry bude zajištěno pomocí úloh definovaných v souboru crontab, který zpracovává a spouští systémový plánovač úloh Cron. Plánovač Cron jsem zvolil z důvodu, že řídící program ze své podstaty nebude napsán jako systémový démon. Tuto funkcionalitu nahrazuje právě Cron, jehož démon crond každou minutu kontroluje příslušné cron tabulky včetně mnou využívané crontab. [12]

2.3.5 Logování

Pro logování chyb použiju systémový logger Rsyslog, který je nejčastějším systémovým logovacím démonem v systémech odvozených od distribuce Debian.

2.3.6 Konfigurační soubor

Pro uložení informací potřebných pro běh zálohování použiji textový soubor s pevně definovanou strukturou a syntaxí, podle které bude program parsovat soubor a načítat potřebné informace.

2.3.6.1 Zabezpečení přístupových údajů

Pro zabezpečení přístupových údajů, které jsou uvedeny v plaintextu v konfiguračním souboru jsem se rozhodl nastavit vlastníka a skupinu konfiguračního souboru na uživatele a skupinu root. Právo na čtení a editaci bude mít pouze uživatel root a skupina root.

2.3.7 Verzovací software

Pro správu verzí uložených konfiguračních souborů a prohlížení rozdílů v konfiguracích jsem se rozhodl neimplementovat vlastní řešení, jelikož tuto úlohu výborně zvládne již existující verzovací systém GIT, který bude volán jako externí program.

2.3.7.1 ***GIT***

GIT je opensource distribuovaný systém pro správu obsahu (SCM). Systém GIT se běžně používá pro správu zdrojových souborů v projektech různorodé velikosti, ale také pro správu textových souborů. [10]

2.4 Systémové a síťové požadavky

2.4.1 Systémové požadavky na server

- Dostatečný diskový prostor pro ukládané konfigurace a binární obrazy IOS dle počtu zálohových zařízení.
- OS linuxová distribuce odvozená od GNU/Linux Debian.
- Funkční TFTP démon s právy na zápis do adresáře určeného pro ukládání záloh.

2.4.2 Síťové požadavky na server

- Povolena odchozí komunikace ze serveru na síťové prvky přes protokol SSH, nebo telnet. Cílové porty síťových prvků TCP 22 a 23.
- Povolena příchozí komunikace ze síťových prvků na server přes protokol TFTP. Cílový port serveru UDP 69.

2.4.3 Požadavky na konfigaraci Cisco IOS

- Vytvořený username s priviledge levelem povolujícím spouštět příkaz copy (Lokální, nebo účet z AAA serveru.).
- Povoleno přihlášení na VTY přes ssh, či telnet s výše uvedeným username.
- Povolena příchozí komunikace ze serveru na TCP porty 22 a 23.
- Povolena odchozí komunikace na UDP port 69 serveru.

Kompatibilita programu je zajištěna pro verze Cisco IOS ve verzích minimálně:

- 12.3(7)T
- 12.2(25)S 12.3(14)T 12.2(27)SBC
- 12.2(31)SB2
- 12.2(33)SRA
- 12.2(33)SXH
- 12.2(33)SB

3 Realizace

3.1 Řídící program

Řídící program ciscoback je po instalaci umístěn v adresáři /opt/ciscoback.

Na následujících řádcích popíšu ukázky kódu z nejdůležitějších funkcí uvnitř zdrojového kódu a blíže rozeberu především princip automatizované komunikace, implementovaný pomocí rozšíření Expect. Kód nedůležitý pro prezentaci bude nahrazen třemi tečkami (…). U funkcí budu uvádět číslo řádku, na němž ve zdrojovém kódu začíná. Zdrojový kód je dostupný v příloze B.

3.1.1 Princip automatizace pomocí Expect

Expect na základě popisu dialogu mezi uživatelem a interaktivním programem umožňuje komunikovat s programem neinteraktivně. Jádrem Expectu je stejnojmenný příkaz **expect**, který popisuje vzor, či seznam vzorů očekávané na výstupu interaktivního programu. Pokud se výstup z interaktivního programu shoduje se vzorem, pak je vykonána příslušná akce.

Ke spoluprácí s příkazem expect využívám příkazy **spawn** a **send**. Příkazem spawn se spustí interaktivní program, jehož výstupy bude očekávat příkaz expect. Pokud se bude vzor shodovat s výstupem programu, pak příkazem send zašlu příkaz interaktivnímu programu. Příkazem send jsou posílány příkazy tak, jako by je zadával uživatel interaktivně přímo v textové konzoli.

3.1.1.1 Ukázka automatizace přihlášení přes telnet a zadání příkazu

Spuštění telnetu ke vzdálenému zařízení. Za příkazem spawn následuje syntaxe programu telnet:

```
spawn telnet 192.168.10.1
```

Očekávání výstupu shodující se s jedním vzorem ze seznamu. Při vypršení časového limitu je vykonán příkaz u vzoru timeout. Pokud bude na výstupu programu telnet řetězec "connected", pak je příkazem send odesláno programu telnet řetězec uvedený mezi uvozovkami ("\r" značí Escape sekvenci Carriage Return.):

```
expect {
      timeout { puts "Vyprsel casovy limit!" }
      "username:" { send "admin\r" }
}
expect "password:"
send "SecretPassword\r"
```

Z výše uvedeného vyplývá, že při implementaci programu budu muset zajistit očekávání správných vzorů odpovídajících všem možných odpovědím interaktivních programů (ssh, telnet apod.) a taky všem možným odpovědím během interakce s OS Cisco IOS.

3.1.2 Funkce pro připojení k síťovému zařízení

Řádek č. 400 v příloze B.

Funkce slouží pro vytvoření připojení k zařízení, přihlášení do zařízení a vrácení id relace. Přijímá jako parametry ip adresu, metodou připojení, login a heslo:

```
proc connect { device method user pass } {
```

…

O spuštění ssh, či telnet připojení se stará komponenta spawn z rozšíření Expect:

```
if { $method == "ssh" } {
        eval spawn ssh -o StrictHostKeyChecking=no $user@$device
      } else {
        eval spawn telnet $device
}
```

Po vyvolání spawnu se testují případné chyby jako vypršení času připojování, neexistující cesta v síti apod. Pokud je vše v pořádku, pak expect očekává na konzoli síťového prvku řetězec "Password:". Následně se příkazem send odesílá login uložený v proměnné. Stejný postup pokračuje u dotazu na heslo.

```
expect {
      timeout {
            log "warning" "Connection to $device timed out!"
            puts "Connection to $device timed out!"
            return 1
      }
            …
            "*No route to host*" {
                  log "error" "No route to $device!"
                  …
                  return 1
            }
            "*Password required, but none set*" {
```

```
                log "warning" "$device user/password not set in
                IOS!"; return 1 }
            "Password:" {send "$pass\r"}
            "Username:"
        }
        send "$user\r"
        expect {
            "Password:"      {send "$pass\r"}
            "Login invalid" {
                …
                return 1
            }
        }
        …
        return $spawn_id
}
```

Pokud se provede úspěšné přihlášení a zařízení se nachází v enable režimu (Expect očekává řetězec obsahující "#".), pak funkce connect vrací id, které identifikuje relaci vytvořenou voláním funkce spawn. Díky tomuto id budou moci další funkce uvnitř skriptu navázat na vytvořenou relaci a dále komunikovat se síťovým zařízením a využívat kombinaci příkazů expect a send.

3.1.3 Funkce pro vyvolání zálohy na tftp

Řádek č. 475 v příloze B.

Funkce slouží pro vyvolání zálohy zvoleného typu konfigurace na tftp server. Vstupními parametry této funkce jsou id relace vytvořené spawnem, ip adresa zařízení, typ zálohované konfigurace, login, heslo, ip adresa tftp serveru a cesta pro uložení:

```
proc backupConfig { id device config user pass tftp_ip
config_path }
```

…

Pro navázání na spawnem vytvořenou relaci z funkce connect je třeba přiřadit do proměnné spawn_id id vrácené funkcí connect:

```
set spawn_id $id
```

…

Expect očekává, že zařízení je stále v enable režimu, a tudíž očekává na konzoli zařízení řetězec obsahující "#". Pokud tomu tak je, pak dojde ke zjištění, jaký typ konfigurace má být zálohován a následnému zaslání sekvence příkazů pro vyvolání zálohy do tftp.

```
expect "#"
…
if { [lsearch $config start] >= 0} {
  …
  send "copy startup-config tftp:\r"
  expect "Address or name of remote host []"
  send "$tftp_ip\r"
  expect "Destination filename"
  set timeout 20
  send "ciscoback/$config_path/${device}_start\r"
  …
  expect {
    "*(Timed out)" { log "warning" "Cannot open tftp session to
    $tftp_ip"; return 1}
    "#" { log "info" "$device backup OK" }
    "Error opening * (Undefined error)" { log "warning" "Cannot
    write to $tftp_ip. Do I have access rights?"; return 1}
  }
…
```

3.1.4 Funkce pro GIT add a commit

Řádek č. 379 v příloze B.

Funkce gitAdd slouží k přidání sledování souboru a vytvoření commitu pro právě odzálohovaný konfigurační soubor v adresáři spravovaným pod GITem. Přijímá jako parametry cestu k adresáři, do něhož se zálohoval konfigurační soubor a ip adresu zařízení, která identifikuje název souboru.

```
proc gitAdd { git_path device }
```

Pokud dojde k nalezení souboru ve tvaru IpAdresa_start, či IpAdresa_run, pak je spawnován program git s parametrem commit.

```
if { [file exists ${device}_start ] } {
```

```
    exec git add ${device}_start
    spawn git commit ${device}_start -m "Ciscoback backup of
    $device startup-config"
    expect {
       "nothing to commit (working directory clean)" {send "\r" }
    }
}
if { [file exists ${device}_run] } {
  exec git add ${device}_run
  …
  }
```

3.1.5 Funkce pro rollback konfigurace

Řádek č. 65 v příloze B.

Funkce rollback slouží k obnovení konfigurace z poslední verze zálohovaného running-config, či startup-config konfiguračního souboru a okamžitému aplikování do running-config síťového zařízení. Při této operaci vyvolávám v IOSu příkaz configure replace, který provádí porovnání běžící konfigurace se souborem určeným k obnově a následně zapisuje pouze změny vyplývající z rozdílů v konfiguraci. Nedochází tedy k pouhému sloučení dvou verzí konfiguračních souborů, nýbrž k přesnému přenastavení aktuální running configurace. Celá operace se může provádět v několika cyklech, aby se správně ošetřily závislosti příkazů.

V některých případech může dojít ke komplikacím při změně nastavení a IOS se poté na tuto skutečnost interaktivně dotazuje. Příkladem komplikace může být např. zrušení nastavení aktivního DHCP serveru na zařízení při procesu obnovení. Nejsem schopen programem ošetřit reakce na veškeré komplikace a správně logicky rozhodnout v dané situaci, tudíž v takovém případě obnova selže a program zaloguje chybu.

Jako vstupní parametry přijímá funkce id relace vytvořené spawnem, cestu k obnovovanému konfiguračnímu souboru a logickou hodnotu parametru write.

```
proc rollback {id path write}
```

V závislosti na parametru write je po úspěšné obnově running-config uloženo na trvalo do startup-config.

…

```
expect "#"
```

```
send "configure replace tftp://$path force\r"
set timeout 5
expect {
      -re "(.*)Could not open file(.*)" { log "error"
      "$expect_out(buffer)!"; return 1 }
      -re "(.*)\nRollback Done\r(.*)" {
            if { $write == 1 } {
                  send "save running-config startup-config\r"
            }
            return 0
      }
      "% Invalid input detected at '^' marker." { return 2 }
}
return 1
...
```

Návratová hodnota 2 je vrácena pokud je příkaz **configure replace** zavolán na zařízení s verzí Cisco IOS, která tuto syntaxi nepodporuje.

3.1.6 Funkce pro kontrolu syntaxe konfiguračního souboru

Řádek č. 99 v příloze B.

Funkce configTest slouží pro kontrolu syntaxe konfiguračního souboru. Parametr isSilent určuje, jestli má program vypisovat případnou chybovou hlášku na konzoli (V případě, že je spuštěn ciscoback ručně s parametrem configtest.).

```
proc configTest { isSilent } {
...
```

Jsou inicializovány výchozí hodnoty parametrů z konfiguračního souboru. Je otevřen a uložen obsah konfiguračního souboru, zjištěny názvy sekcí a uloženy všechny řádky souboru do proměnné:

```
set tftp_ip 0
set tftp_path 0
...
```

```
set lines [openConfig]
set sections [getSections $lines]
set lines [split $lines "\n"]
```

...

Na uložených řádcích je nejprve kontrolována existence výchozích parametrů pomocí regulárních výrazů. V ukázce je uvedena kontrola paramteru ip adresy tftp serveru a cesty k tftp adresáři:

```
foreach line $lines {
    if { [regexp "^\[ \t\]*tftp_ip\[ \t\]*((\[2\]\[5\]\[0-
    5\]|(\[2\]\[0-4\]|\[1\]\[0-9\]|\[0-9\])?\[0-
    9\])\.){3}(\[2\]\[5\]\[0-5\]|(\[2\]\[0-4\]|\[1\]\[0-
    9\]|\[0-9\])?\[0-9\])\[ \t\]*" $line temp] != 0 } {
        set tftp_ip 1
        log "debug" "configtest: tftp_ip found."
    }
    if { [regexp "^\[ \t\]*tftp_path\[ \t\]*\".*\"" $line temp]
    != 0 }{
        set tftp_path 1
        log "debug" "configtest: tftp_path found."
    }
```

...

Následně je kontrolována existence parametrů v jednotlivých sekcích. V ukázce je uvedena kontrola parametru cesty k adresáři pro zálohu konfigurace:

```
set lines [ join $lines "\n" ]
    foreach section $sections {
        if { [regexp  "section\[
        \t\]*\"$section\".*endsection\[ \t\]*\"$section\""
        $lines slines] != 0 } {
                if { [regexp -line "^\[ \t\]*config_path\[
                \t\]*\".*\"" $slines temp] != 0 } {
                    set config_path 1
                    log "debug" "configtest on section
                    $section: config_path found."
                ...
```

Jsou nalezeny a kontrolovány řádky obsahující informace o jednotlivých zařízeních. V ukázce je uvedeno hledání záznamů pro hosty a jejich parametry (Ip adresa, typ zálohy, metoda připojení k zařízení.).

```
set hostlines [ split $slines "\n" ]
      foreach line $hostlines {
      …
            if { [regexp -line "^\[ \t\]*host\[
            \t\]*(((\[2\]\[5\]\[0-5\]|(\[2\]\[0-4\]|\[1\]\[0-
            9\]|\[0-9\])?\[0-9\])\.){3})(\[2\]\[5\]\[0-
            5\]|(\[2\]\[0-4\]|\[1\]\[0-9\]|\[0-9\])?\[0-9\])\[
            \t\]*(\[rsi\]\[ \t\]*)+\[ \t\]*(ssh)|(telnet)\[
            \t\]*$" $line temp] != 0 } {
      …
```

3.1.7 Rozsah zdrojového kódu

Výsledný řídící program obsahuje bez komentářů a prázdných řádků 665 řádků kódu.

3.2 Init skript

Init skript ciscoback splňuje náležitosti LSB (Linux Standard Base) init skriptu. Nachází se ve výchozím init adresáři /etc/init.d.

Ačkoliv neobsluhuje klasického binárního démona, jelikož je ciscoback interpretovaným programem, zvolil jsem jeho použití pro zachování logického uspořádání z pohledu operačnícho systému. V run-levelech 0,1,6 nedochází k ukončení démona, nýbrž k vymazání záznamů v crontabu. Stejně tak v run-levelech 2,3,4,5 dochází při správně nastaveném konfiguračním souboru k zaregistrování úloh do crontabu.

Funkce uvnitř init skriptu volají řídící program ciscoback s danými přepínači. Např. spuštěním init skriptu s parametrem start dojde k otestování, zdali je init skript spuštěn uživatelem s root právy a poté dojde k volání:

```
/opt/ciscoback/ciscoback --save
```

3.3 Konfigurační soubor

Konfigurační soubor lze rozdělit na globální část a sekce. Svou syntaxí umožnuje rozdělit síťové prvky do logických skupin. Logické rozdělení může být dáno buďto stejným nastavením přihlašovacích údajů, nebo je jím možno rozdělit zařízení do skupin podle času

a typu zálohy jaký se má provést. Další možností je logické rozdělení z důvodů administračních (Např. různé geografické lokace, různá politika správy IT apod.).

Globální parametry:

- Ip adresa tftp serveru.
- Cesta k tftp adresáři.
- Výchozí cesta k adresáři s konfiguračními soubory.
- Výchozí cesta k adresáři s IOS obrazy.
- Výchozí čas zálohování konfiguračních souborů.
- Výchozí čas zálohování IOS obrazů.
- Výchozí přihlašovací login.
- Výchozí přihlašovací heslo.

Parametry sekcí:

- Čas zálohování konfiguračních souborů.
- Čas zálohování IOS obrazů.
- Cesta k adresáři s konfiguračními soubory
- Cesta k adresáři s IOS obrazy
- Přihlašovací login.
- Přihlašovací heslo.
- Minimálně jeden záznam popisující zálohované zařízení (Ip adresa, typ zálohy, metoda připojení k zařízení.)

Podrobný popis syntaxe s ukázkami se nachází v uživatelském manuálu (Příloha A.).

3.4 Log

Nastavení logování pro ciscoback v rsyslogu je provedeno v postinst souboru v instalačním balíčku. Cesta k ciscoback logu, do kterého zapisuje rsyslog je:

```
/var/log/ciscoback/ciscoback.log
```

Zapisováno do něj je díky přidání následujícího řádku do /etc/rsyslog.conf.

```
local0.*   /var/log/ciscoback/ciscoback.log
```

3.5 Manuálová stránka

Manuálová stránka Ciscoback je uložena v sekci 1 v adresáři /usr/share/man/man1 a splňuje formát Unixové manuálové stránky.

3.6 Deb balíček

Balíček je závislý na balíčcích tcl8.5, expect, telnet, openssh-client, git, rsyslog a logrotate. Uvnitř balíčku se nachází řídící program, init skript, konfigurační soubor a manuálová stránka. Při instalaci jsou rozkopírovány do příslušných cílových adresářů.

Po instalaci balíčku se provádějí následující akce:

- Nastavení rsyslog facility local0 v rsyslogu.conf a restart služby rsyslog.
- Vytvoření adresářové struktury a souboru pro log.
- Vytvoření záznamu v logrotate.
- Vytvoření softlinků na init skript v příslušných run-level adresářích pomocí update-rc.d.

Po odinstalaci balíčku jsou prováděny následující akce:

- Odebrání softlinků init skriptu z run-level adresářů pomocí update-rc.d
- Smazání záznamů v crontabu.
- Smazání záznamů v logrotate.
- Zrušení ciscoback local0 facility v rsyslog.
- Smazání souborů z balíčku, jež byly nakopírovány při instalaci.

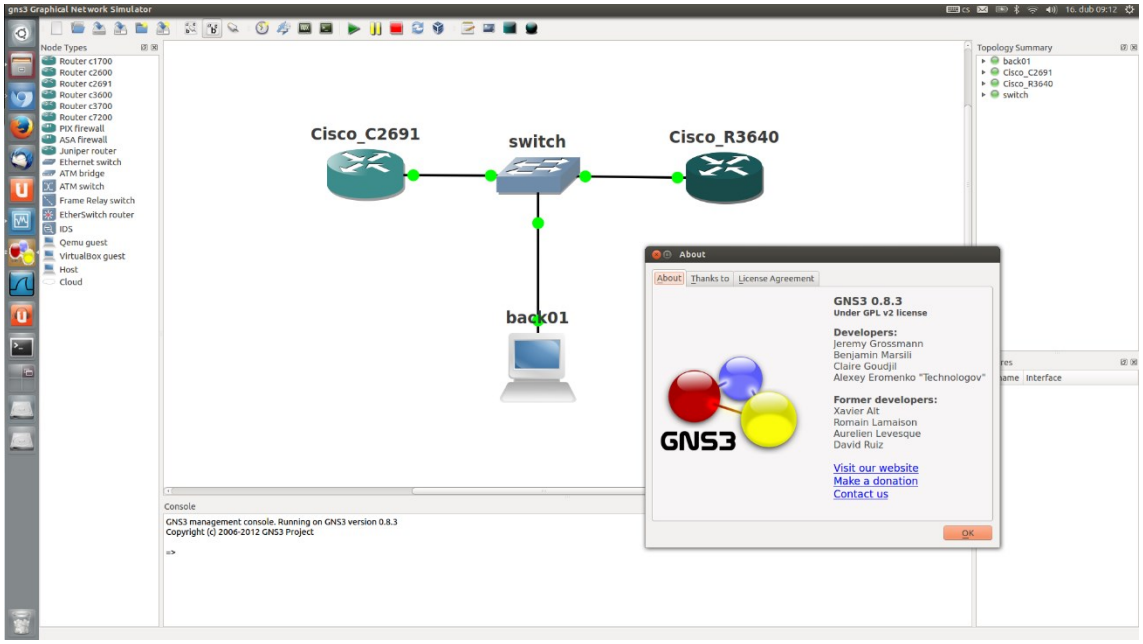
Log soubor je ponechán.

4 Testovací prostředí

Ciscoback jsem otestoval ve virtuálním i fyzickém prostředí. V obou případech jsem testoval celou funkční specifikaci zadání.

4.1 Virtuální prostředí pomocí GNS3

Ciscoback jsem otestoval ve virtuální síťové topologii tvořené programem GNS3 a VirtualBox. Ve Virtualboxu jsem vytvořil virtuální počítač se systémem Debian ve verzi 7.4 (32bit). Tftp server na virtuální OS zprostředkovával software tftpd-hpa. Tento virtuální stroj jsem poté importoval do topologie v GNS3. Jako virtuální Cisco zařízení jsem použil směrovače C2691 a R3640. Schéma topologie je vyznačeno na obrázku 1.1.



Obrázek 1.1: *Virtuální topologie v programu GNS3*

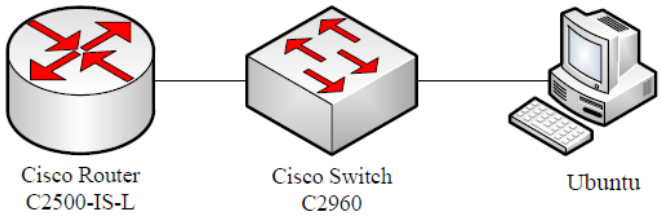
4.1.1 GNS3

GNS3 je open-source grafický software určený pro simulaci komplexních sítí bez použití dedikovaných hardwarových prvků. Simulovaná zařízení běží ve virtuálních strojích. Pro simulaci virtuálního počítače je použit software Virtualbox, či Qemu. Pro simulaci síťových prvků je použit emulátor Dynamips. Díky virtualizaci plnohodnotných zařízení lze simulovat reálnou síťovou topologii. [11]

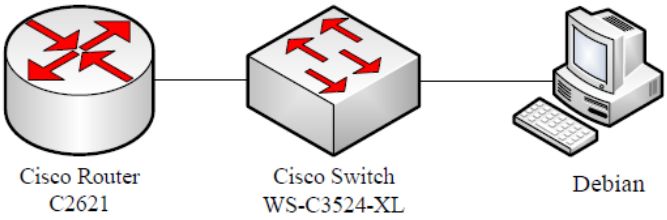
4.2 Testování na fyzických topologiích

Ciscoback jsem testoval také na dvou fyzických topologiích.

Topologie č. 1 je vyobrazena na obrázku 1.2. Topologie č. 2 je vyobrazena na obrázku 1.3. Na obou obrázcích jsou popsány typy reálných prvků. Jako server jsem používal počítače s OS Ubuntu ve verzi 13.04 (64bit) a Debian ve verzi 7.5 (64bit). Na obou OS byl jako tftp server použit software tftpd-hpa.



Obrázek 1.2: *Fyzická topologie č. 1*



Obrázek 1.3: *Fyzická topologie č. 2*

5 Dostupnost

Ciscoback je šířen pod licencí GNU GPL v3 a je veřejně dostupný z webu:

https://sourceforge.net/projects/ciscoback/

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat systém pro údržbu záloh konfiguračních souborů síťových zařízení. Jako cílovou platformu síťových zařízení jsem si vybral enterprise modely síťových prvků společnosti Cisco, Inc., na nichž se nachází operační systém Cisco IOS. Mezi další požadavky mimo zálohu konfiguračních souborů patřilo také zálohování obrazu operačního systému, možnost obnovy konfigurace zařízení a navrhnutí systému pro verzování a porovnávání obsahu konfiguračních souborů.

Po analýze a srovnání již existujících řešení od výrobce zařízení i softwaru jiných dodavatelů, jež se zaobírají touto problematikou, jsem navrhnul své vlastní řešení. Mé řešení vzešlo z potřeby vytvořit systém na míru síťové topologii ve společnosti, v níž vykonávám funkci síťového administrátora. Nicméně řešení jsem chtěl navrhnout natolik obecně, aby jej mohl použít kdokoliv jiný se sítí, v níž se nachází síťové prvky Cisco s Cisco IOS. Nosnou myšlenkou mého řešení je využití kombinace TCL skriptu s rozšířením o Expect, jenž automatizuje interaktivní komunikaci se síťovými prvky.

V průběhu práce jsem se setkal s několika problémy v implementaci a v testování. Problémem při tetování bylo problematické zprovoznění virtuální topologie v software GNS3, kdy jsem měl často problém zprovoznit komunikaci mezi virtuálním pc a virtuálními síťovými prvky. Jednodušší bylo provádět testy na fyzické topologii. Jako problém v implementaci vidím např. problematiku logování. Nejsem spokojen s absencí možnosti vytvořit si vlastní logovací facility s názvem aplikace ciscoback. Musel jsem v Rsyslogu využít facility local0.

Oficiální požadavky na mou práci byly naplněny. Nicméně stále existuje velký prostor pro zlepšení funkčnosti systému. Mezi ně patří především možnost paralelismu při větším množství prvků, možnost výběru obnovovaného konfiguračního souboru podle data, nikoliv pouze poslední zálohy. Dále je otázka zdali je volba zabezpečení přístupových údajů dostatečná a jestli by ji nebylo vhodné přeimplementovat jinak. V další verzi by také bylo vhodné rozšířit tento systém o možnost zálohování i jiných enterprise, či manažovatelných modelů síťových prvků jiných výrobců (Checkpoint, Juniper, HP, D-Link atd.) a také vytvořit RPM balíček Ciscobacku pro operační systémy odvozené od distribuce Red Hat Enterpise Linux.

Práce pro mne byla přínosem, jelikož jsem si rozšířil znalosti o zálohovacích mechanismech a možnostech automatizace s jazykem TCL s rozšířením Expect. Díky tvorbě instalovatelného deb balíčku a logickému začlenění do systému na bázi Debianu, jsem si také osvojil principy instalací, služeb, logování a plánování v operačních systémech GNU/Linux.

Použitá literatura

[1] Evi, NEMETH, Garth SNYDER a Trent R. HEIN. Linux: kompletní příručka administrátora. 1.vyd. Brno: Computer Press, 2004, 828 s. ISBN 80-722-6919-4

[2] WRZESZCZ, Zdisław. TCL/TK: podrobný průvodce programovacími jazyky. Vydání první. Brno: Computer Press, 2005, 388 s. ISBN 80-251-0781-7

[3] LIBES, Don. Exploring expect: a tcl-based toolkit for automating interactive programs. 1st ed. Sebastopol, CA: O'Reilly, c1995, xxxiii, 566 p. ISBN 15-659-2090-2

[4] Cisco IOS and NX-OS Software. CISCO SYSTEMS, Inc. Cisco IOS and NX-OS Software [online]. 2014 [cit. 2014-04-01]. Dostupné z: http://www.cisco.com/c/en/us/products/ios-nx-os-software/index.html

[5] BOUŠKA, Petr. Www.samuraj-cz.com [online]. 2014 [cit. 2014-04-01]. Dostupné z: http://www.samuraj-cz.com/clanky-kategorie/cisco-admin/

[6] Solarwinds - Network Configuration Manager. SOLARWINDS, Inc. Solarwinds - Network Configuration Manager [online]. 2014 [cit. 2014-04-01]. Dostupné z: http://www.solarwinds.com/network-configuration-manager.aspx

[7] Command Scheduler (Kron). In: *Cisco Networking Services Configuration Guide, Cisco IOS Release 15M&T* [online]. 170 West Tasman Drive, San Jose, CA 95134-1706, 2012 [cit. 2014-04-1]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cns/configuration/15-mt/cns-15-mt-book/cns-cmd-sched.html

[8] RANCID - Really Awesome New Cisco confIg Differ. SHRUBBERY NETWORKS, Inc. Http://www.shrubbery.net/ [online]. 2006 [cit. 2014-04-01]. Dostupné z: http://www.shrubbery.net/rancid/

[9] DEBIAN. Debian [online]. 2013 [cit. 2014-04-01]. Dostupné z: http://www.debian.org/

[10] GIT: --distributed-is-the-new-centralized [online]. 2014 [cit. 2014-04-1]. Dostupné z: http://git-scm.com/

[11] GNS3: Graphical Network Simulator [online]. 2014 [cit. 2014-04-1]. Dostupné z: http://www.gns3.net/

[12] VÁCLAVÍK, Jiří. Linux v příkazech - plánované spouštění procesů. In: Linuxsoft.cz [online]. 2006 [cit. 2014-04-1]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1178#cron

Seznam příloh na CD

Příloha A: Uživatelský manuál .....CD/uzivatelsky_manual.pdf

Příloha B: Zdrojový kód řídícího programu ciscoback .....CD/zdrojovy_kod

Příloha C: Instalovatelný deb balíček…….CD/ciscoback_1.0.deb